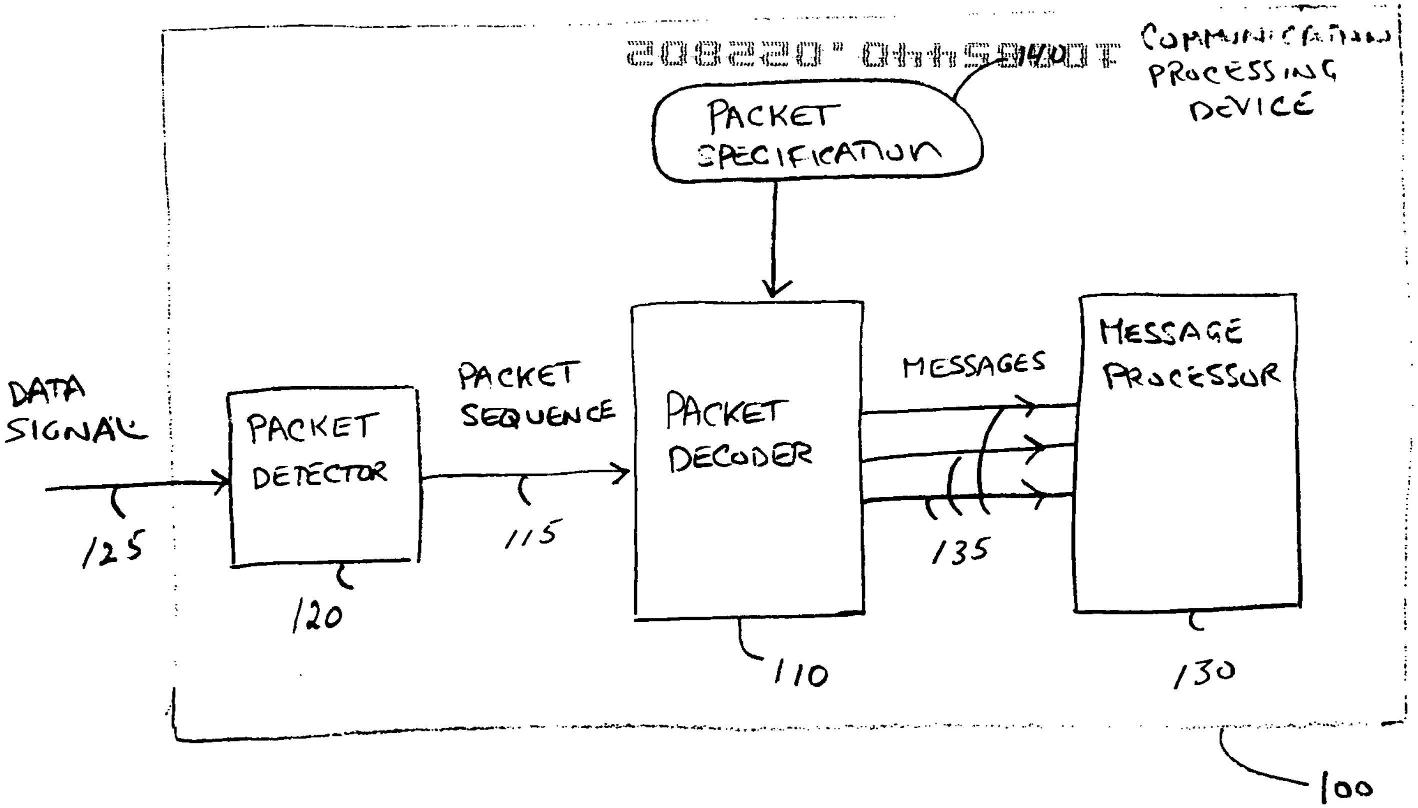
COMMUNICATION PROCESSING DEVICE
PACKET SPECIFICATION
140
DATA SIGNAL
125
PACKET DETECTOR
120
PACKET SEQUENCE
115
PACKET DECODER
110
MESSAGES
135
MESSAGE PROCESSOR
130
100

FIG. 1

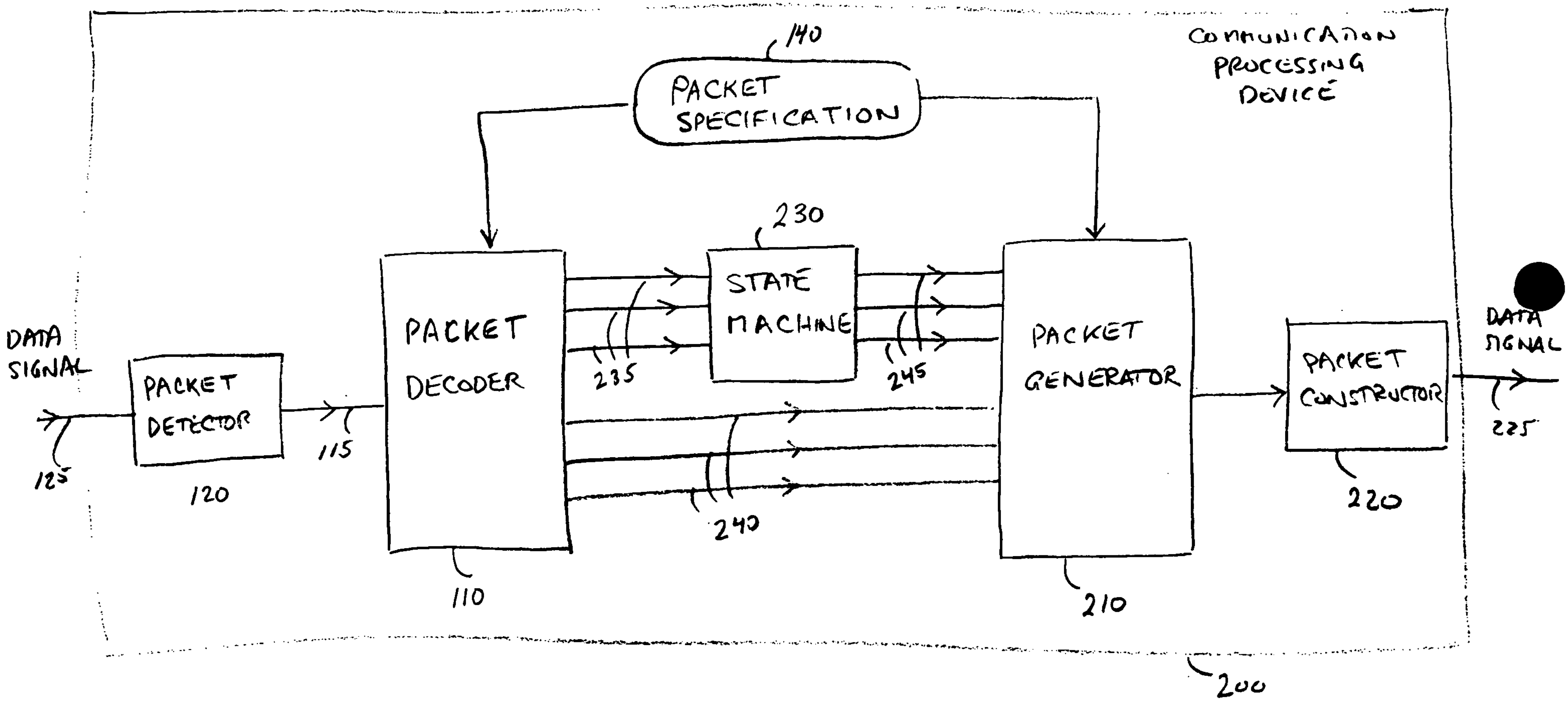
COMMUNICATION
PROCESSING
DEVICE
200
DATA
SIGNAL
125
PACKET
DETECTOR
120
115
PACKET
DECODER
110
140
PACKET
SPECIFICATION
230
STATE
MACHINE
235
245
240
PACKET
GENERATOR
210
PACKET
CONSTRUCTOR
220
DATA
SIGNAL
225

FIG. 2

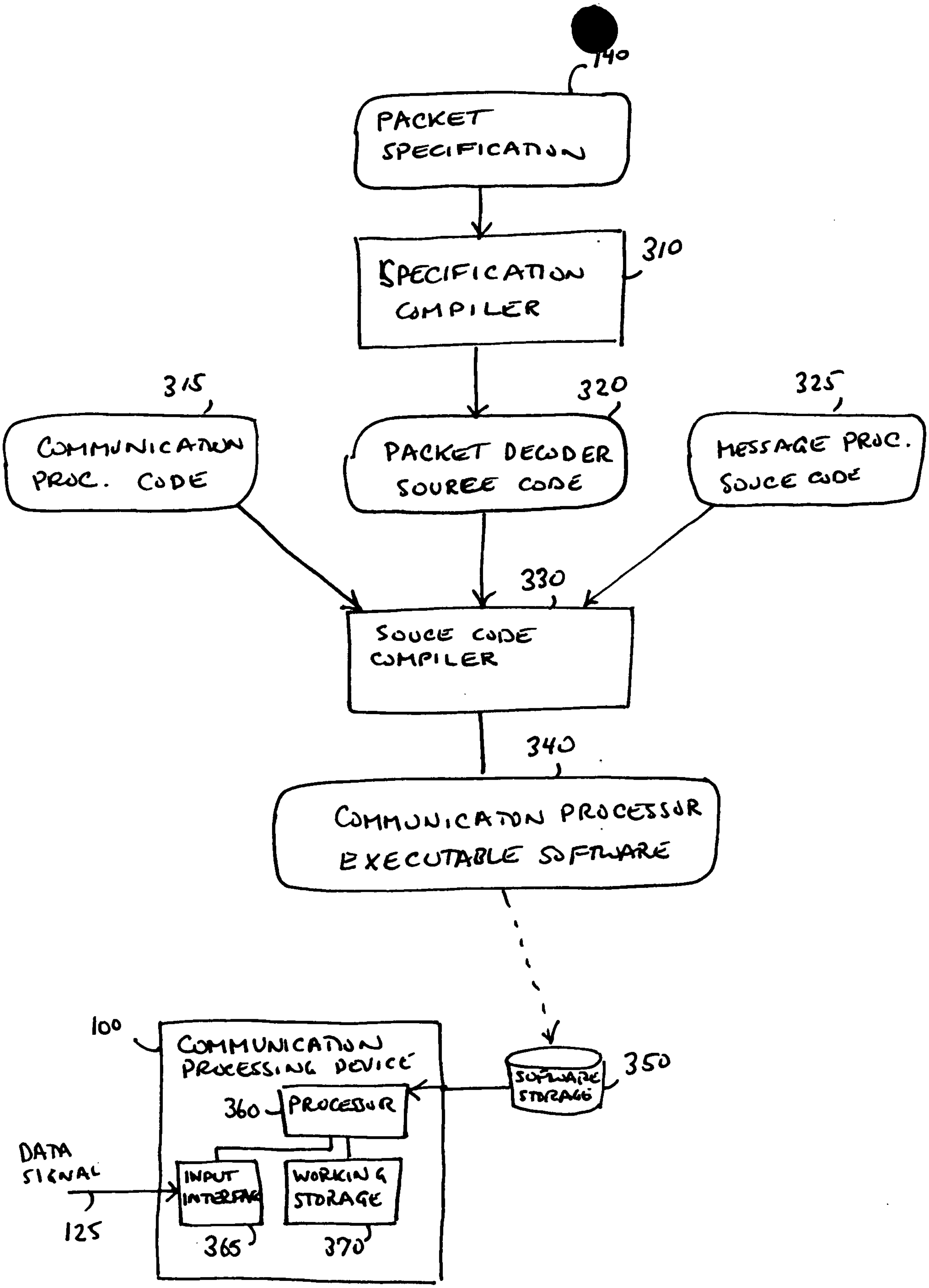
140
PACKET SPECIFICATION
SPECIFICATION COMPILER
310
315
COMMUNICATION PROC. CODE
320
PACKET DECODER SOUREE CODE
325
MESSAGE PROC. SOUCE CODE
330
SOUCE CODE COMPILER
340
COMMUNICATION PROCESSOR EXECUTABLE SOFTWARE
100
COMMUNICATION PROCESSING DEVICE
360
PROCESSOR
350
SOFTWARE STORAGE
DATA SIGNAL
125
INPUT INTERFACE
WORKING STORAGE
365
370

FIG. 3

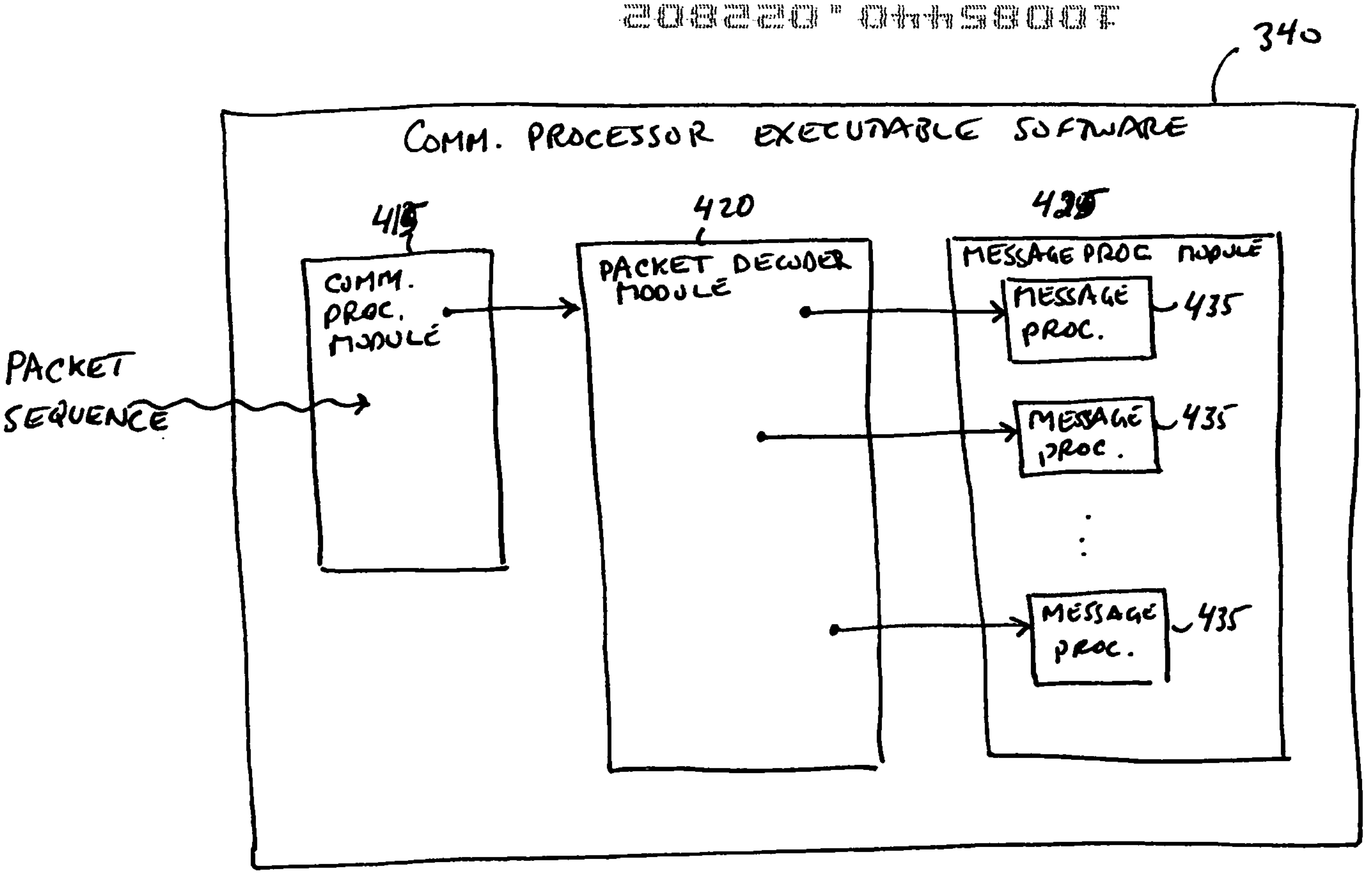
340
COMM. PROCESSOR EXECUTABLE SOFTWARE
415
COMM. PROC. MODULE
PACKET SEQUENCE
420
PACKET DECODER MODULE
425
MESSAGE PROC MODULE
MESSAGE PROC.
435
MESSAGE PROC.
435
MESSAGE PROC.
435

FIG. 4

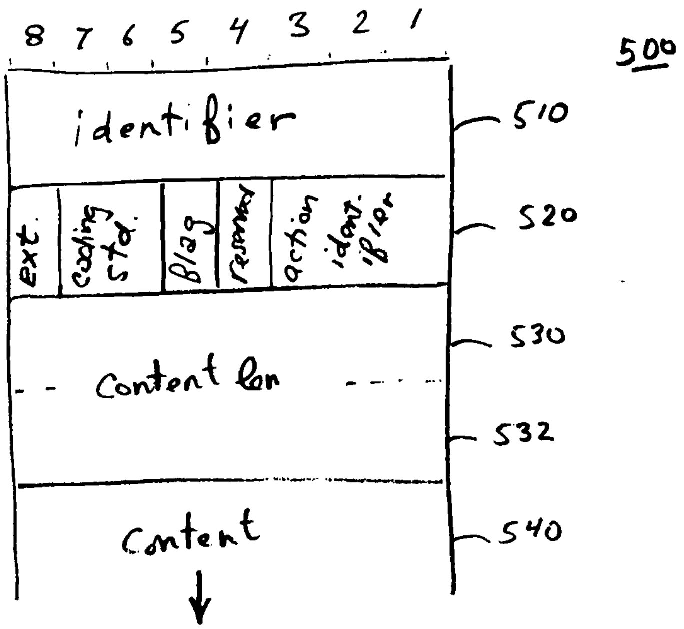
8 7 6 5 4 3 2 1
500
identifier
510
ext.
coding std.
flag
reserved
action identifier
520
Content len
530
532
Content
540

FIG. 5

```
packet ("information element") {
   field(:"identifier",8) {alt{
     0x04:"narrow-band bearer capability"
     0x08:"cause" 0x14:"call state"
     0x1e:"progress indicator"
     .
     .
     .
     0x5c:"quality of service parameter"
     0x5e:"broad-band bearer capability"
     0x5f:"broad-band low layer information"
     .
     .
     .
     0x7c:"narrow-band low layer compatibility"
     0x7d:"narrow-band high layer compatibility"
     rest:"unknown"}
   }
   field(:"ext bit")
   field(:"coding standard",2) {alt{
     0:"ITU-T standartized coding"
     1:"ISO/IEC standard"
     2:"national standard"
     3:"standard defined for the network present on the network instruction field is
     not significant"}
   }
   field(:"flag")
   field(:"reserved") {"should be 0x00"}
   field(:"action identifier",3)
   len field("contens len",16)
```

FIG. 6A

```
alt(var(:"identifier")) {
      0x04:packet(:"narrow-band bearer capability")         // Q.2931 pg. 79
                                                            // Q.2931 pg. 64
      0x08:packet(:"cause")                                 // Q.2931 pg. 68
      0x14:packet(:"call state")                            // Q.2931 pg. 59
      0x1e:packet(:"progress indicator")                    // Q.2931 pg. 81
.
.
.
      0x5c:packet(:"quality of service parameter")          // Q.2931 pg. 72
      0x5e:packet(:"broad-band bearer capability")          // Q.2931 pg. 51
      0x5f:packet(:"broad-band low layer information")      // Q.2931 pg. 54
.
.
.
      0x7c:packet(:"narrow-band low layer compatibility")   // Q.2931 pg. 80
      0x7d:packet(:"narrow-band high layer compatibility")  // Q.2931 pg. 79
      rest:"unknown"
   }
}
```

FIG. 6B

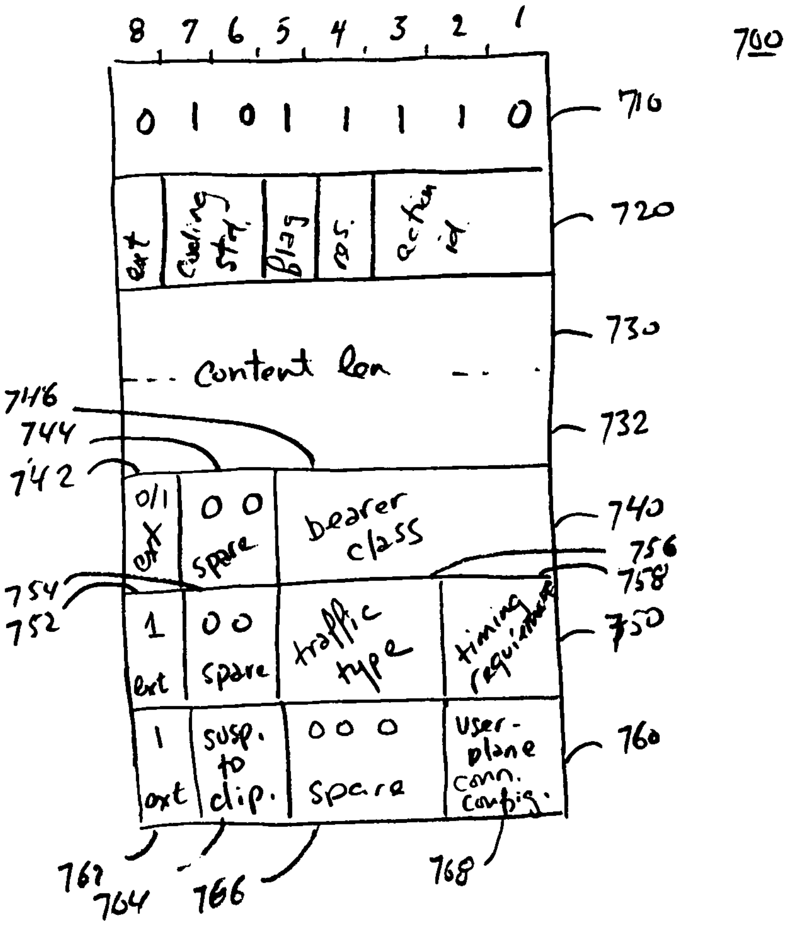
700
8 7 6 5 4 3 2 1
0 1 0 1 1 1 1 0
710
ext
coding std.
flag
res.
action id
720
... content len ...
730
732
746
744
742
0/1
ext
0 0
spare
bearer class
740
756
758
754
752
1
ext
0 0
spare
traffic type
timing requirement
750
1
ext
susp. to clip.
0 0 0
spare
User-plane conn. config.
760
762
764
766
768

FIG. 7

```
packet("broad-band bearer capability") {
 field(:"ext bit")
 field(:"spare", 2)
 field("bearer class",5) {alt{
  1:"BCOB-A" 3:"BCOB-C"  0x10:"BCOB-X" rest:"reserved"}}
 alt(var(:"ext bit")) {
  0: field(:"ext bit") field(:"spare",2)
   field("traffic type",3) {alt{
    0:"no indication"
    1:"constant bit rate"
    2:"variable bit rate"
    rest:"reserved"}}
  field("timing requirements",2) {alt{
    0:"no indication"
    1:"end-to-end timing required"
    2:"end to end timing not required"
    rest:"reserved"}}
 }
 field(:"ext bit")
 field("susceptibility to clipping",2) {alt{
  0:"not susceptible to clipping"
  1:"susceptible to clipping"
  rest:"reserved"}}
 field(:"spare",3)
 field("user plane connection configuration",2) {alt{
  0:"point-to-point"
  1:"point-to-multipoint"
  rest:"reserved"}}
}
```

FIG. 8

```
void C$Q_2931_Signaling::PP$information_element (char* aname, unsigned a) {
 if( !CurPduLen() ) return;
 ProcessPacketBeg( aname, a );

 V$FP$identifier = ProcessField ( "identifier", 8);
 FP$identifier ( V$FP$identifier );
 V$FP$ext_bit = ProcessField ( "ext bit", 1);
 FP$ext_bit ( V$FP$ext_bit );
 V$FP$coding_standard = ProcessField ( "coding standard", 2);
 FP$coding_standard ( V$FP$coding_standard );
 V$FP$flag = ProcessField ( "flag", 1);
 FP$flag ( V$FP$flag );
 V$FP$reserved = ProcessField ( "reserved", 1);
 FP$reserved ( V$FP$reserved );
 V$FP$action_identifier = ProcessField ( "action identifier", 3);
 FP$action_identifier ( V$FP$action_identifier );
 V$FP$information_element$contens_len = ProcessField ( "contens len", 16);
 FP$information_element$contens_len ( V$FP$information_element$contens_len );
 SetPduLen ( V$FP$information_element$contens_len );

 if ( V$FP$identifier == 0x04 ) {
  PP$narrow_band_bearer_capability ( "narrow-band bearer capability", CurPduLen());

 } else
 if ( V$FP$identifier == 0x08 ) {
  PP$cause ( "cause", CurPduLen());

 } else
 if ( V$FP$identifier == 0x14 ) {
  PP$call_state ( "call state", CurPduLen());

```

FIG. 9A

```
  } else
  if ( V$FP$identifier == 0x1e ) {
   PP$progress_indicator ( "progress indicator", CurPduLen());

 .
 .
 .

  } else
  if ( V$FP$identifier == 0x5c ) {
   PP$quality_of_service_parameter ( "quality of service parameter", CurPduLen());

  } else
  if ( V$FP$identifier == 0x5e ) {
   PP$broad_band_bearer_capability ( "broad-band bearer capability", CurPduLen());

  } else
  if ( V$FP$identifier == 0x5f ) {
   PP$broad_band_low_layer_information ( "broad-band low layer information",
 CurPduLen());

 .
 .
 .

  } else
  if ( V$FP$identifier == 0x7c ) {
   PP$narrow_band_low_layer_compatibility ( "narrow-band low layer compatibility",
 CurPduLen());

```

FIG. 9B

```
  } else
  if ( V$FP$identifier == 0x7d ) {
   PP$narrow_band_high_layer_compatibility ( "narrow-band high layer compatibility",
CurPduLen());

  } else  {
   PrintName("unknown");
  } ;
  ProcessPacketEnd(aname, a );
} // end of packet processor function definition PP$information_element
```

FIG. 9C

```
void C$Q_2931_Signaling::FP$identifier (unsigned a) {

 PrintName(": ");

 if ( a == 0x04 ) {
  PrintName("narrow-band bearer capability");
 } else
 if ( a == 0x08 ) {
  PrintName("cause");
 } else
 if ( a == 0x14 ) {
  PrintName("call state");
 } else
 if ( a == 0x1e ) {
  PrintName("progress indicator");

.
.
.

 } else
 if ( a == 0x5c ) {
  PrintName("quality of service parameter");
 } else
 if ( a == 0x5e ) {
  PrintName("broad-band bearer capability");
 } else
 if ( a == 0x5f ) {
  PrintName("broad-band low layer information");

```

FIG. 9D

```
  .
  .
  .

   } else
   if ( a == 0x7c ) {
    PrintName("narrow-band low layer compatibility");
   } else
   if ( a == 0x7d ) {
    PrintName("narrow-band high layer compatibility");
   } else  {
    PrintName("unknown");
   } ;
  } // end of field processor function definition FP$identifier
```

FIG. 9E

```
void C$Q_2931_Signaling::PP$broad_band_bearer_capability (char* aname, unsigned a) {
 if( !CurPduLen() ) return;
 ProcessPacketBeg( aname, a );

 V$FP$ext_bit = ProcessField ( "ext bit", 1);
 FP$ext_bit ( V$FP$ext_bit );
 V$FP$spare = ProcessField ( "spare", 2);
 FP$spare ( V$FP$spare );
 V$FP$broad_band_bearer_capability$bearer_class = ProcessField ( "bearer class", 5);
 FP$broad_band_bearer_capability$bearer_class (
V$FP$broad_band_bearer_capability$bearer_class );

 if ( V$FP$ext_bit == 0 ) {
 V$FP$ext_bit = ProcessField ( "ext bit", 1);
 FP$ext_bit ( V$FP$ext_bit );
 V$FP$spare = ProcessField ( "spare", 2);
 FP$spare ( V$FP$spare );
 V$FP$broad_band_bearer_capability$traffic_type = ProcessField ( "traffic type", 3);
 FP$broad_band_bearer_capability$traffic_type (
              V$FP$broad_band_bearer_capability$traffic_type );
 V$FP$broad_band_bearer_capability$timing_requirements = ProcessField ( "timing
requirements", 2);
 FP$broad_band_bearer_capability$timing_requirements (
V$FP$broad_band_bearer_capability$timing_requirements );

 } else  ;
   V$FP$ext_bit = ProcessField ( "ext bit", 1);
 FP$ext_bit ( V$FP$ext_bit );
 V$FP$broad_band_bearer_capability$susceptibility_to_clipping = ProcessField (
"susceptibility to clipping", 2);
```

FIG. 9F

```
 FP$broad_band_bearer_capability$susceptibility_to_clipping (
V$FP$broad_band_bearer_capability$susceptibility_to_clipping );
 V$FP$spare = ProcessField ( "spare", 3);
 FP$spare ( V$FP$spare );
 V$FP$broad_band_bearer_capability$user_plane_connection_configuration =
ProcessField ( "user plane connection configuration", 2);
 FP$broad_band_bearer_capability$user_plane_connection_configuration
         (V$FP$broad_band_bearer_capability$user_plane_connection_configuration );

 ProcessPacketEnd(aname, a );
}
// end of packet processor function definition PP$broad_band_bearer_capability
```

FIG. 9G

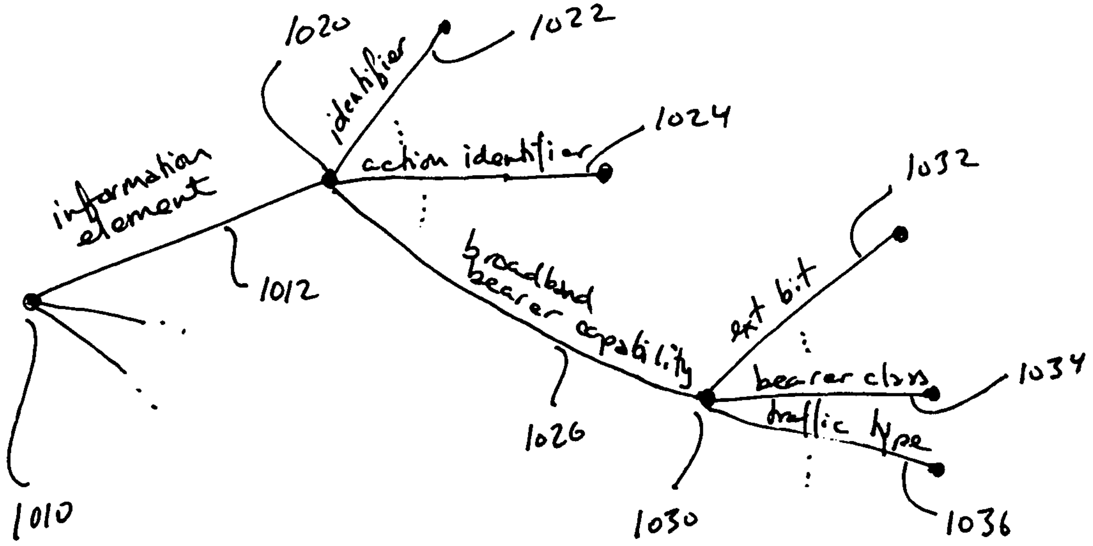
information element
1010
1012
1020
identifier
1022
action identifier
1024
broadband bearer capability
1026
1030
ext bit
1032
bearer class
1034
traffic type
1036

FIG. 10

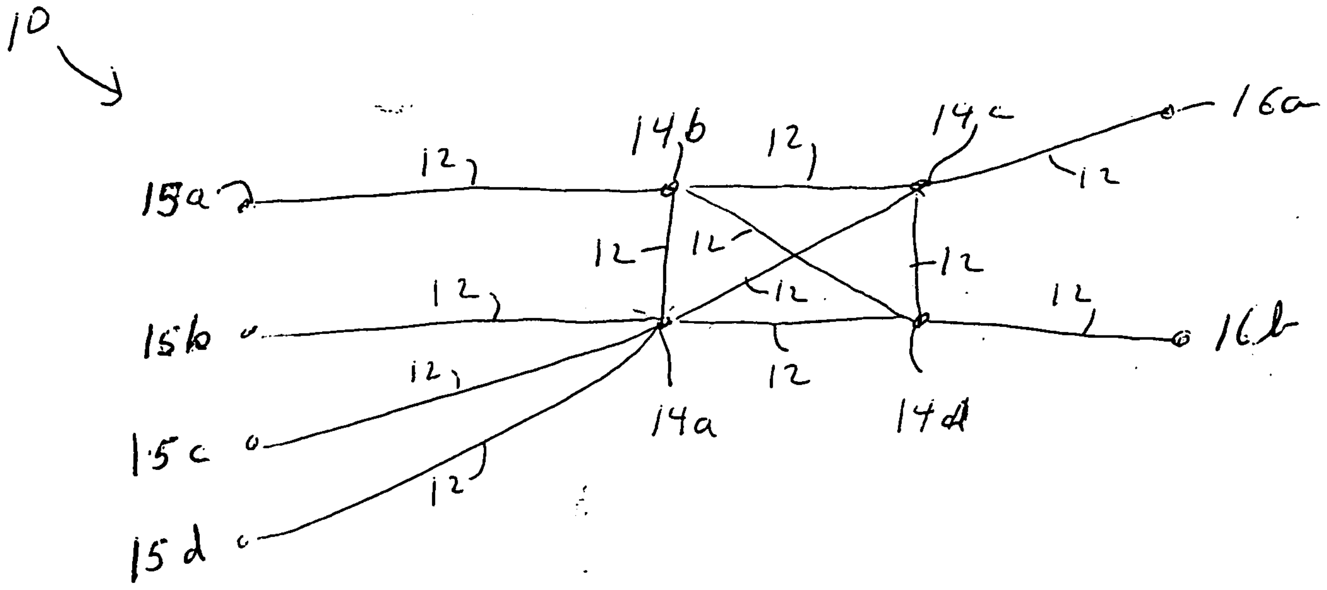

Fig. 1

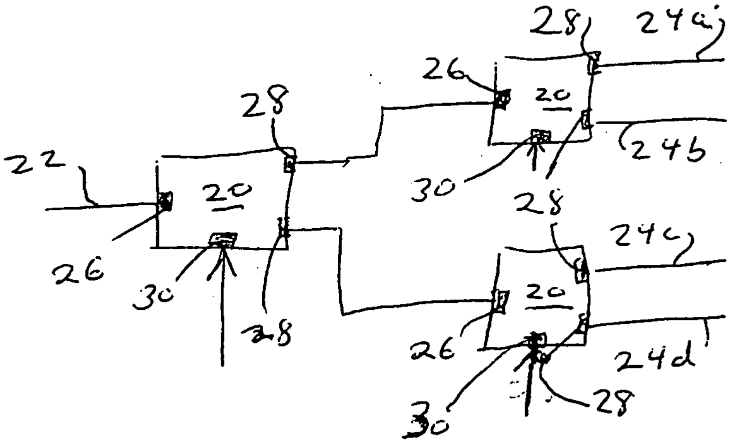

Fig. 2

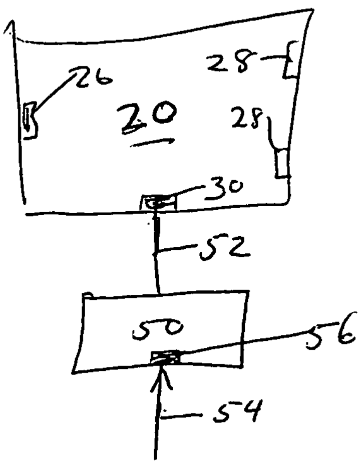

Fig. 3

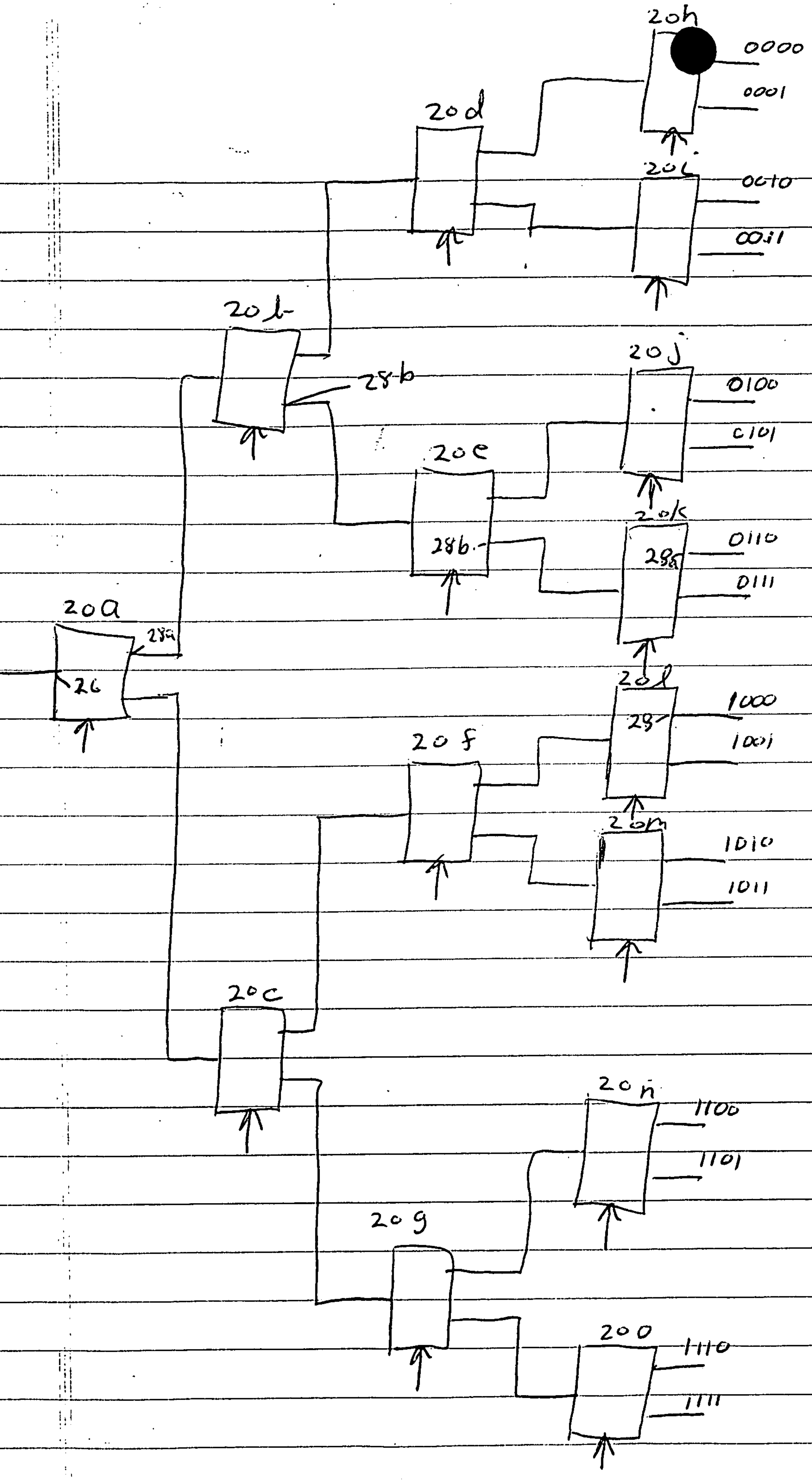
20h
0000
0001
20d
20i
0010
0011
20b
28b
20j
0100
0101
20e
28b
20k
28a
0110
0111
20a
28a
26
20l
28
1000
1001
20f
20m
1010
1011
20c
20n
1100
1101
20g
20o
1110
1111

Figure 4

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0000	1	1	x	1	x	x	x	1	x	x	x	x	x	x	x
0001	1	1	x	1	x	x	x	0	x	x	x	x	x	x	x
0010	1	1	x	0	x	x	x	x	1	x	x	x	x	x	x
0011	1	1	x	0	x	x	x	x	0	x	x	x	x	x	x
0100	1	0	x	x	1	x	x	x	x	1	x	x	x	x	x
0101	1	0	x	x	1	x	x	x	x	0	x	x	x	x	x
0110	1	0	x	x	0	x	x	x	x	x	1	x	x	x	x
0111	1	0	x	x	0	x	x	x	x	x	0	x	x	x	x
1000	0	x	1	x	x	1	x	x	x	x	x	1	x	x	x
1001	0	x	1	x	x	1	x	x	x	x	x	0	x	x	x
1010	0	x	1	x	x	0	x	x	x	x	x	x	1	x	x
1011	0	x	1	x	x	0	x	x	x	x	x	x	0	x	x
1100	0	x	0	x	x	x	1	x	x	x	x	x	x	1	x
1101	0	x	0	x	x	x	1	x	x	x	x	x	x	0	x
1110	0	x	0	x	x	x	0	x	x	x	x	x	x	x	1
1111	0	x	0	x	x	x	0	x	x	x	x	x	x	x	0

← 54

"1" - top port
"0" - bottom port
"x" - "don't care"

Figure 5

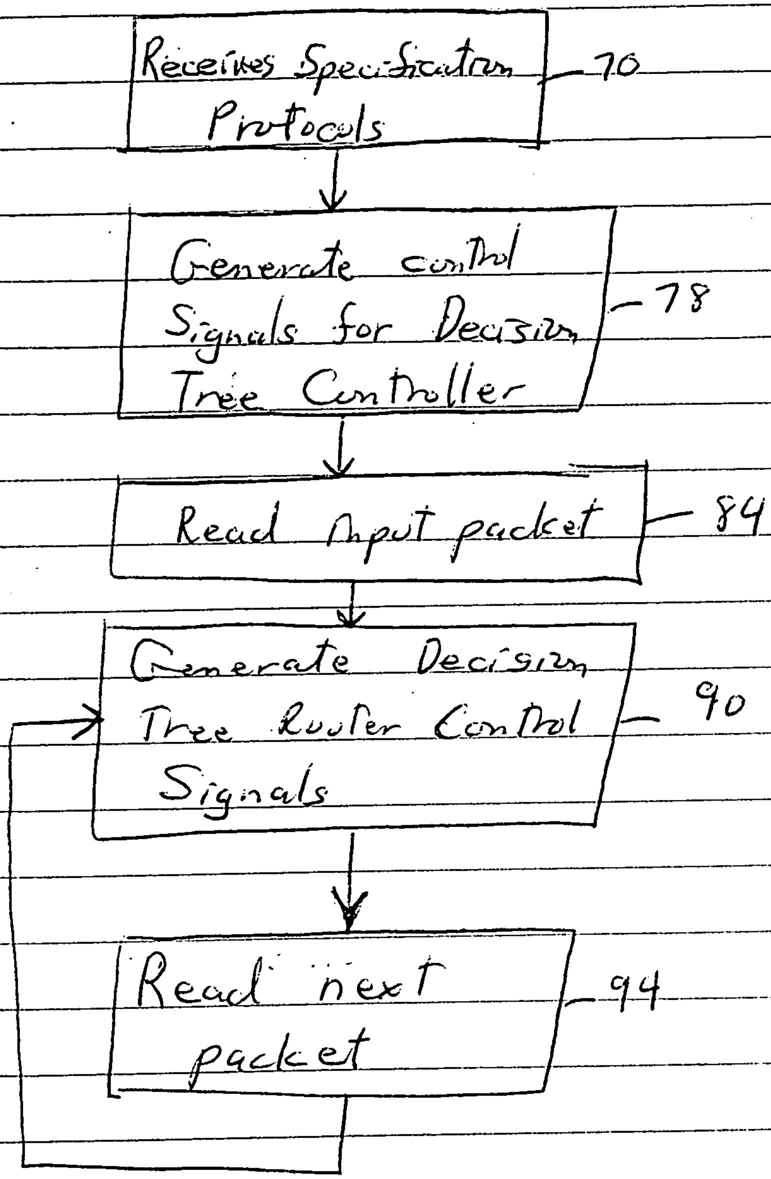
Receives Specification Protocols
70
Generate control Signals for Decision Tree Controller
78
Read Input packet
84
Generate Decision Tree Router Control Signals
90
Read next packet
94

Fig 6

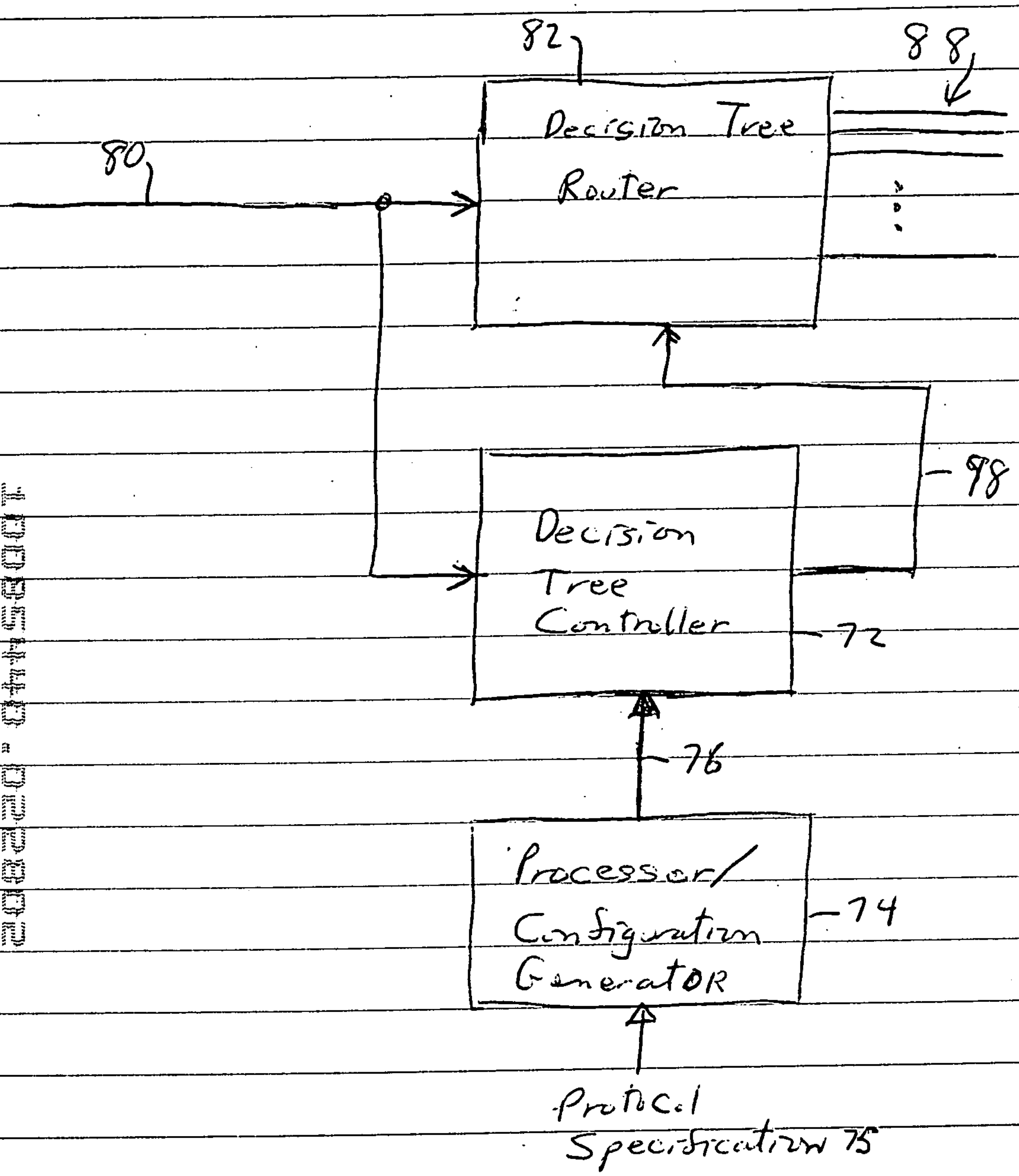
82
88
Decision Tree
Router
80
98
Decision
Tree
Controller
72
76
Processor/
Configuration
GeneratOR
74
Protocol
Specification 75

Fig. 7